EEE4120F Practical 1

Vikyle Naidoo[†] and Anoubhav Pudaruth[‡]
EEE4120F Class of 2020
University of Cape Town
South Africa
[†]NDXVIK005 [‡]PDRANO001

***Abstract*—Performance measurement using OCTAVE; an introduction to benchmarking, the golden measure, speed-up and report writing.**

I. Introduction

In this practical we will investigate and compare the performance of OCTAVE code for various implementations of the same functions, both built-in and user defined. We will compare the performance of rand() when using the matrix functionality as opposed to using for-loops to generate the same amount of white noise.

By using the built-in correlation function as a golden measure we can determine the speedup of our user defined correlation function and hence compare the performance.

We will also investigate the effects of correlation on identical but shifted signals.

II. Methodology

A. Hardware

These experiments were run on an i7-7500U CPU.

B. Experimental Procedure

In the first experiment, we generated 10s of white noise using the matrix functionality of rand(), and the wavwrite() function. We compared the speed of this to the speed of generating the same white noise using a for-loop and generating a single random noise value on every iteration. The hypothesis here was the the loop iteration method would be much slower than the matrix implementation.

In the second experiment, we compared the performance of OCTAVE's built-in correlation function to our user defined correlation function based on Pearson's formula. We generated white noise of varying sample sizes and calculated the speedup between the two function for each one. Here we speculated that our implementation would give similar outputs as the built in function but would run faster since our implementation is quite simple.

For the last experiment, we compared signals shifted in time. We generated sin curves of varying frequency and sampling sizes, and compared samples of the same sizes that are shifted in time. We theorised that, for a fixed sample shift, the correlation between the two signals would increase as the sample size increased and frequency decreased.

C. Implementation

For the first experiment we generated white noise as follows:

```
function whiten = createwhiten(n)
  whiten = [];
  for i = 1:n*8000
    r = rand()*2-1;
    %whiten = vertcat(whiten, r);
    whiten = [whiten;r];
  endfor
 return;
endfunction
```

This was compared to the standard way of generating a random matrix, which we used as our golden measure for this experiment:

```
rand(48000*10, 1)*2 - 1
```

and the execution time of each was measured using the tic and toc functions.

In the next experiment, we implemented a correlation function using Pearson's formula as follows:

```
%input two matrices x and y; output correlation coefficient r
% I am assuming that the inputs are both 1D matrices (vectors) of the same length
function r = mycorr(x, y)
  n = size(x);
  n = n(2);
  x_mean = mean(x);
  y_mean = mean(y);

  x_diff = x-x_mean;
  y_diff = y-y_mean;

  numerator = sum((x_diff).*(y_diff));
  denominator = (sqrt(sumsq(x_diff))).*(sqrt(sumsq(y_diff)));
  r = numerator./denominator;

  return;
endfunction
```

The output of our correlation function was then compared to that of Octave's built-in correlation function using the following code:

```
%comparing the outputs of mycorr and corr

[x, fs] = audioread('w.wav');
y = x;
r1 = mycorr(x,y)
r2 = corr(x,y) # note that in some versions, this is called "corr"
disp(r2 - r1);

y(1) = 2; y(5) = -4; # i.e. fudge some of the value
r1 = mycorr(x,y)
r2 = corr(x,y)
disp(r2 - r1);

x = rand(1,10); y = rand(1,10);
r1 = mycorr(x,y)
r2 = corr(x,y)
disp(r2 - r1);
```

The average execution speed of both functions were compared as well. Different sample sizes (100, 1000 & 10000) were used and for each of them, both functions were run 3 times to obtain an average execution time as shown in the code below.

```
%comparing average execution speeds of mycorr and corr

x = rand(1,1000); y = rand(1,1000);

tic; r1 = mycorr(x,y); runtime = toc();
tic; r1 = mycorr(x,y); runtime1 = toc();
tic; r1 = mycorr(x,y); runtime2 = toc();
tic; r1 = mycorr(x,y); runtime3 = toc();
runtime4 = (runtime1+runtime2+runtime3)/3; %average runtime
disp(strcat("It took: ", num2str(runtime4*1000), " ms to run mycorr."));

tic; r2 = corr(x,y); runtime = toc();
tic; r2 = corr(x,y); runtime1 = toc();
tic; r2 = corr(x,y); runtime2 = toc();
tic; r2 = corr(x,y); runtime3 = toc();
runtime4 = (runtime1+runtime2+runtime3)/3; %average runtime
disp(strcat("It took: ", num2str(runtime4*1000), " ms to run corr."));
disp(r2 - r1);
```

Finally, the correlation of identical but shifted signals was calculated using the built-in corr function. Sine curves of varying frequency and sampling sizes (100,1000 and 10000) were created and each time, one of the two signals was shifted in time by 10 samples. The code used for the creation and correlation of the signals is shown below.

```
%correlation of shifted signals (shifted by 10 samples)

n = 0:0.001:100;
w = pi/10;

n1 = n(1:10001);
n2 = n(11:10011);

a = sin(w*n1);
b = sin(w*n2);

subplot(2,1,1)
stem (n1,a)
title('Plot of A and B (Shifted by 10 samples) for \omega=\pi/10')
xlabel('n1')
ylabel('a = sin(\pi/10*n1)')
subplot(2,1,2)
stem (n2,b)
xlabel('n2')
ylabel('b = sin(\pi/10*n2)')
r = corr(a,b)
```

III. Results

A. Random White Noise Generator

Upon calculation of the speedup, our function, createwhiten, performed considerably slower than the golden measure, with speedup of 0.05% . This may be explained by the fact that Octave is optimised for vectorised calculations, which make use of parallelism, but by using a for-loop which is a sequential operation, it takes longer.

The results however were accurate and gave the same white noise as expected from the golden measure, a histogram of which is presented below.

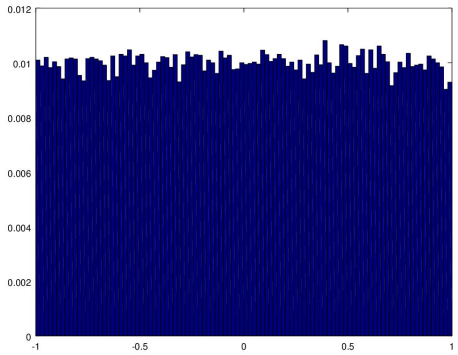

Fig. 1: Histogram of white noise generate by createwhiten(N)

B. Correlation Function

Comparing the output of our correlation function to the built-in one, it was found that both functions gave the same result with differences occurring in the order of magnitude less than 10^{-10}.

As for the execution speed, the mycorr function was faster than the built-in function for all sample sizes. This was expected since our function made use of simple calculations while OCTAVE's built in function may be doing more complex calculations. The results obtained are shown in Table I.

TABLE I

TABLE OF SPEED-UP VS SAMPLE SIZE FOR MYCORR FUNCTION

Sample Size	mycorr time/ ms	corr time/ ms	Speed-up
100	0.8973	3.0207	3.366
1000	0.9460	2.9194	3.086
10000	0.9323	3.2154	3.449

C. Correlation of Shifted Signals

For this comparison, the results obtained are shown in Table II below.

As can be seen, the correlation value of shifted signals are less than 1, which means that they do not completely match. However, it can be observed that as frequency decreased and sample size increased, the correlation value became closer to one, which is what we expected.

Some of the shifted signals compared can be seen in Figures 2, 3 and 4 below.

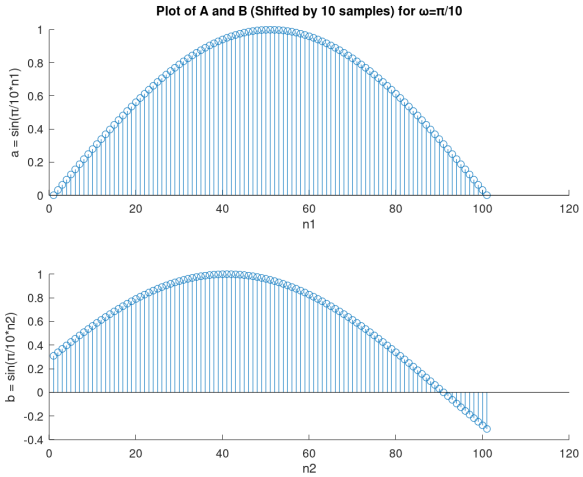

Fig. 2: Plot of Shifted Signals for $\omega = \pi/10$ (sample size = 100)

TABLE II
CORRELATION VALUES FOR DIFFERENT SAMPLE SIZES AND FREQUENCIES

Sample Size	Frequency		
	$\pi/10$	$2\pi/5$	$4\pi/5$
100	0.80452	0.30629	-0.80626
1000	0.99741	0.9921	0.96852
10000	0.99997	0.99992	0.99968

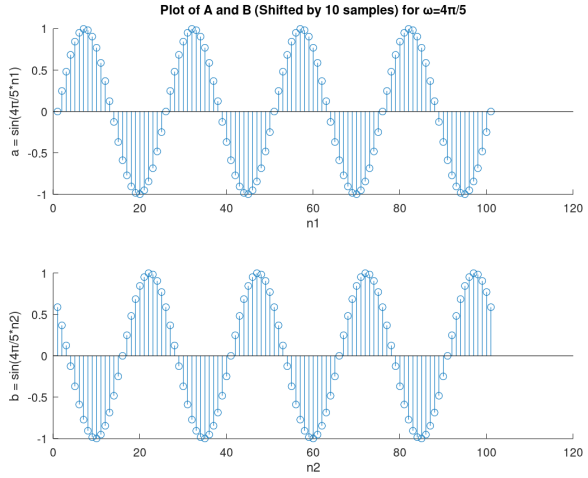

Fig. 3: Plot of Shifted Signals for $\omega = 4\pi/5$ (sample size = 100)

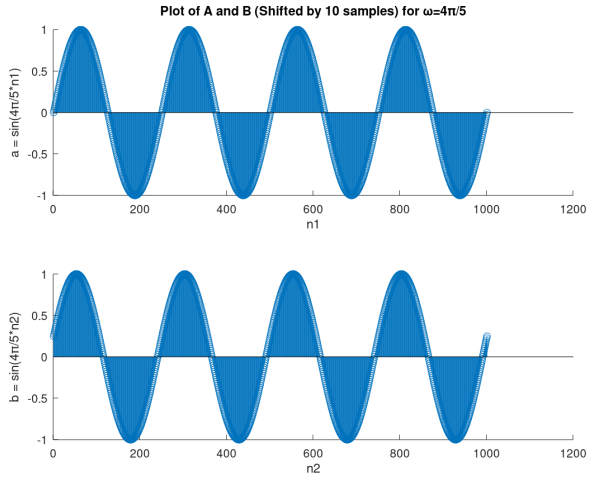

Fig. 4: Plot of Shifted Signals for $\omega = 4\pi/5$ (sample size = 1000)

IV. CONCLUSION

The first experiment lead to the conclusion that using a for-loop to generate a list of random numbers was less efficient than the vectorised generation of the same data, due to the sequential nature of the for-loop, resulting in the compiler not being able to use parallelism. This was a significant reduction in performance with a speedup of 0.05%.

In the second experiment, we came to the conclusion that our user defined implementation of the correlation function actually performed better than the builtin Octave correlation function, with a speedup of up to 340% and negligible accuracy loss. This was anticipated due to our simpler implementation.

In the final experiment we concluded that the correlation coefficient between two shifted signals improved as frequency decreased and sample size increased.